L 12

```
   SSSSSSSSSSSS  YYY         YYY     SSSSSSSSSSSS
   SSSSSSSSSSSS  YYY         YYY     SSSSSSSSSSSS
   SSSSSSSSSSSS  YYY         YYY     SSSSSSSSSSSS
SSS              YYY         YYY  SSS
SSS              YYY         YYY  SSS
SSS              YYY         YYY  SSS
SSS                 YYY   YYY     SSS
SSS                 YYY   YYY     SSS
SSS                 YYY   YYY     SSS
   SSSSSSSSS           YYY           SSSSSSSSS
   SSSSSSSSS           YYY           SSSSSSSSS
   SSSSSSSSS           YYY           SSSSSSSSS
            SSS        YYY                    SSS
            SSS        YYY                    SSS
            SSS        YYY                    SSS
            SSS        YYY                    SSS
            SSS        YYY                    SSS
            SSS        YYY                    SSS
SSSSSSSSSSSS           YYY        SSSSSSSSSSSS
SSSSSSSSSSSS           YYY        SSSSSSSSSSSS
SSSSSSSSSSSS           YYY        SSSSSSSSSSSS
```

```
**FILE**ID**SYSFAO                                          F  7




  SSSSSSSS  YY      YY    SSSSSSSS  FFFFFFFFFF    AAAAAA      OOOOOO
  SSSSSSSS  YY      YY    SSSSSSSS  FFFFFFFFFF    AAAAAA      OOOOOO
SS          YY      YY  SS          FF          AA      AA  OO      OO
SS          YY      YY  SS          FF          AA      AA  OO      OO
SS            YY  YY    SS          FF          AA      AA  OO      OO
SS            YY  YY    SS          FF          AA      AA  OO      OO
  SSSSSS        YY        SSSSSS    FFFFFFFF    AA      AA  OO      OO
  SSSSSS        YY        SSSSSS    FFFFFFFF    AA      AA  OO      OO
        SS      YY              SS  FF          AAAAAAAAAA  OO      OO
        SS      YY              SS  FF          AAAAAAAAAA  OO      OO
        SS      YY              SS  FF          AA      AA  OO      OO      ....
        SS      YY              SS  FF          AA      AA  OO      OO      ....
SSSSSSSS        YY      SSSSSSSS    FF          AA      AA    OOOOOO        ....
SSSSSSSS        YY      SSSSSSSS    FF          AA      AA    OOOOOO        ....

LL            IIIIII      SSSSSSSS
LL            IIIIII      SSSSSSSS
LL              II      SS
LL              II      SS
LL              II      SS
LL              II      SS
LL              II        SSSSSS
LL              II        SSSSSS
LL              II              SS
LL              II              SS
LL              II              SS
LL              II              SS
LLLLLLLLLL    IIIIII    SSSSSSSS
LLLLLLLLLL    IIIIII    SSSSSSSS
```

Table of contents

```
0000          .TITLE  SYSFAO - FORMATTED ASCII OUTPUT SYSTEM SERVICE
0000          .IDENT  'V04-000'
0000    ;
0000    ;****************************************************************************
0000    ;*                                                                          *
0000    ;*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                 *
0000    ;*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                  *
0000    ;*  ALL RIGHTS RESERVED.                                                    *
0000    ;*                                                                          *
0000    ;*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED   *
0000    ;*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE   *
0000    ;*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER   *
0000    ;*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY   *
0000    ;*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY   *
0000    ;*  TRANSFERRED.                                                            *
0000    ;*                                                                          *
0000    ;*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE   *
0000    ;*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT   *
0000    ;*  CORPORATION.                                                            *
0000    ;*                                                                          *
0000    ;*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS   *
0000    ;*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                 *
0000    ;*                                                                          *
0000    ;*                                                                          *
0000    ;****************************************************************************
0000
0000    ;++
0000    ; FACILITY: SYSTEM SERVICE
0000    ;
0000    ; ABSTRACT:
0000    ;
0000    ;       This module provides general formatting services.  It converts
0000    ;       binary values to octal, hexadecimal, and decimal ASCII
0000    ;       representations, and also inserts ASCII strings and converts
0000    ;       date and time to ASCII.
0000    ;
0000    ; ENVIRONMENT:
0000    ;
0000    ;       FAO runs in the mode of the caller.
0000    ;
0000    ; AUTHOR: Henry M. Levy , CREATION DATE: 29-JAN-1977
0000    ;
0000    ; MODIFIED BY:
0000    ;
0000    ;       V03-014 LJK0278         Lawrence J. Kenah       2-May-1984
0000    ;               Move this code to separate program section to reduce the
0000    ;               strain on the cursed word displacements.
0000    ;
0000    ;       V03-013 LMP0201         L. Mark Pilant,         28-Feb-1984  13:22
0000    ;               Add support for formatting the match-all identifier.
0000    ;
0000    ;       V03-012 LMP0169         L. Mark Pilant,         11-Nov-1983  15:07
0000    ;               Correctly handle member wildcards in the %I directive.
0000    ;
0000    ;       V03-011 LMP0119         L. Mark Pilant,         16-Jun-1983  12:05
0000    ;               Make non-translating identifiers appear as hex numbers.
0000    ;
```

```
0000    ;       V03-010 JLV0257         Jake VanNoy             23-MAY-1983
0000    ;               Change !AF to not make "." out of valid 8 bit characters.
0000    ;
0000    ;       V03-009 LMP0111         L. Mark Pilant,         9-May-1983  9:45
0000    ;               Add a new directive, %I, to allow formatting of identifiers.
0000    ;
0000    ;       V03-008 LMP0078         L. Mark Pilant,         10-Feb-1983  12:52
0000    ;               Modify the method used when checking for wildcard group
0000    ;               and member portions of the UIC.
0000    ;
0000    ;       V03-007 LMP0056         L. Mark Pilant,         28-Oct-1982  20:50
0000    ;               Correct a problem introduced by LMP0052 which caused a
0000    ;               truncated search of the % directive table.
0000    ;
0000    ;       V03-006 LMP0052         L. Mark Pilant,         14-Oct-1982  12:30
0000    ;               Add a new directive, !%U, to allow formatting of a UIC.
0000    ;
0000    ;       V03-005 MSH0001         Maryann S. Hinden       20-NOV-1981
0000    ;               Use longword displacement to reference EXE$SIGTORET.
0000    ;
0000    ;       V03-004 DWT0001         David W. Thiel          06-Nov-1981
0000    ;               Fixed condition handler.  Check argument to $ASCTIM to
0000    ;               prevent exception in $ASCTIM.
0000    ;
0000    ;       V03-003 PCA0001         Paul C. Anagnostopoulos         22-Jul-1981
0000    ;               Fixed a bug wherein !AF did not replace unprintable
0000    ;               characters if it encountered result string overflow.
0000    ;               Now it replaces those characters that it does copy.
0000    ;
0000    ;       V03-002 TCM0001         Trudy C. Matthews       10-Mar-1981
0000    ;               Change CALLS with word displacement to CALLS with longword
0000    ;               displacement.
0000    ;
0000    ;       V03-001 TMH0001         Tim Halvorsen   24-Feb-1981
0000    ;               Add condition handler to catch access violations
0000    ;               and the like, so that services like $PUTMSG do
0000    ;               not cause an access violation in programs like DCL
0000    ;               simply because not enough arguments were supplied.
0000    ;--
```

```
                                  0000          .SBTTL  DECLARATIONS
                                  0000
                                  0000  ;
                                  0000  ; MACROS:
                                  0000  ;
                                  0000
                                  0000          $SSDEF                          ; define system status codes
                                  0000          $CHFDEF                         ; Condition handling facility
                                  0000          $SFDEF                          ; Call frame definitions
                                  0000          $UICDEF                         ; UIC FIELD OFFSETS
                                  0000
                                  0000  ;
                                  0000  ; EQUATED SYMBOLS:
                                  0000  ;
                                  0000
                        00000000  0000          ARGCOUNT = 0                    ; offset to argument count
                        00000004  0000          INDSC    = 4                    ; offset to input string descriptor
                        00000008  0000          OUTLEN   = 8                    ; offset to output length
                        0000000C  0000          OUTDSC   = 12                   ; offset to output buffer descriptor
                        00000010  0000          FIRSTARG = 16                   ; offset to first conversion param
                                  0000
                        FFFFFFF0  0000          INLEN    = -16                  ; local offset to input length remaining
                        FFFFFFF4  0000          INPTR    = -12                  ; local offset to input string pointer
                        FFFFFFF8  0000          LASTVAL  = -8                   ; local offset to last value converted
                        FFFFFFFC  0000          FIELDEND = -4                   ; local offset to end of defined field
                                  0000
                        0000000D  0000          CR       = 13                   ; carriage return
                        0000000A  0000          LF       = 10                   ; line feed
                        00000021  0000          EXCL     = 33                   ; exclamation ('!')
                        00000009  0000          TAB      = 9                    ; horizontal tab
                        0000000C  0000          FF       = 12                   ; form feed
                                  0000
                                  0000  ;
                                  0000  ; OWN STORAGE:
                                  0000  ;
                                  0000
                        00000000          .PSECT  YF$SYSFAO
                                  0000
                                  0000  ASC_NAMES:
42 41 39 38 37 36 35 34 33 32 31 30  0000          .ASCII  /0123456789ABCDEF/      ; ASCII digits
                         46 45 44 43  000C
                                  0010
                                  0010  ;
                                  0010  ;       The following table contains the first character for all
                                  0010  ;       FAO conversion directives.  The first part of the table
                                  0010  ;       contains the first character for two-character directives,
                                  0010  ;       while the second half of the table contains the one-character
                                  0010  ;       directives.
                                  0010  ;
                                  0010  ;       NOTE -- The ordering of this table must be preserved.  The index
                                  0010  ;               of the directives found in this table is used to dispatch
                                  0010  ;               via a CASE statement in the main program (FAO).
                                  0010  ;               Routine CVTNUM also uses the index to dispatch and to
                                  0010  ;               compute the proper radix for the conversion.
                                  0010  ;
                                  0010
                                  0010  CNTRL_TABLE:
```

```
                 0010  TWO_CHAR_CNTRLS:
             4F  0010          .ASCII  /O/                     ; octal conversions
             58  0011          .ASCII  /X/                     ; hex conversions
             55  0012          .ASCII  /U/                     ; unsigned decimal
             53  0013          .ASCII  /S/                     ; signed decimal
             5A  0014          .ASCII  /Z/                     ; unsigned decimal zero filled
             41  0015          .ASCII  /A/                     ; ascii insertion directives
             25  0016          .ASCII  /%/                     ; time conversion, plural indication, or  UI
             2A  0017          .ASCII  /*/                     ; character repeater
                 0018  ONE_CHAR_CNTRLS:
             2B  0018          .ASCII  /+/                     ; skip argument
             2D  0019          .ASCII  /-/                     ; backup argument
             3C  001A          .ASCII  /</                     ; begin field definition
             3E  001B          .ASCII  />/                     ; end of field definition
                 001C  REPLACE_CHRS:                           ; these are one or two char replacements
             2F  001C          .ASCII  ./.                     ; newline
             5F  001D          .ASCII  /_/                     ; tab
             5E  001E          .ASCII  /^/                     ; form feed
             21  001F          .ASCII  /!/                     ; insert exclamation
       00000010  0020  CNTRL_LENGTH = .-CNTRL_TABLE            ; length of table
                 0020
       00000008  0020  ONECHAR_INDEX = CNTRL_LENGTH - <ONE_CHAR_CNTRLS - CNTRL_TABLE>
                 0020
       0000000C  0020  REPL_OFFSET = REPLACE_CHRS - CNTRL_TABLE ; offset of replacement chars
                 0020
                 0020  STRING_TYPES:
    46 44 53 43  0020          .ASCII  /CSDF/                  ; ascii string types
                 0024  DATA_TYPES:
       4C 57 42  0024          .ASCII  /BWL/                   ; byte, word , or long
                 0027  PERCENT_STR:
 54 44 53 49 55  0027          .ASCII  /UISDT/                 ; subtypes for % directive
                 002C  FIELDS:
       20 10 08  002C          .BYTE   8,16,32                 ; field size for B,W,and L
                 002F  REPLACEMENT:
    21 0C 09 0A  002F          .BYTE   LF,TAB,FF,EXCL          ; simple replacement table
                 0033
                 0033  ;
                 0033  ; The following array contains the number of Octal and Hex digits in
                 0033  ; byte , word, and longword fields.  The byte digits are first, the
                 0033  ; hex digits starting at the 4'th entry so that the array may be
                 0033  ; context indexed.
                 0033  ;
                 0033
                 0033  OCT_HEX_DIGITS:
    00 0B 06 03  0033          .BYTE   3,6,11,0
       08 04 02  0037          .BYTE   2,4,8
                 003A
                 003A  RADIX:
 0A 0A 0A 10 08  003A          .BYTE   8,16,10,10,10           ; radix for numeric conversisons
                 003F
                 003F
```

```
003F          .SBTTL  FAO - MAIN PROGRAM
003F    ;++
003F    ; FUNCTIONAL DESCRIPTION:
003F    ;
003F    ;       This routine is the entry point for the FAO and FAOL system
003F    ;       services.  The caller's control string is scanned for control
003F    ;       characters ('!').  All other information is simply passed to
003F    ;       the output buffer.  If a control directive is found, it is parsed
003F    ;       and an action routine is cispatched.
003F    ;
003F    ; CALLING SEQUENCE:
003F    ;
003F    ; CALLS or CALLG        to SYS$FAO or SYS$FAOL
003F    ;
003F    ; INPUT PARAMETERS:
003F    ;
003F    ;       INDSC    - The address of a string descriptor for the input
003F    ;                  control string.
003F    ;       OUTLEN   - The address of a word to receive the length of
003F    ;                  the output string
003F    ;       OUTDSC   - The address of a string descriptor for the output
003F    ;                  buffer.
003F    ;       FIRSTARG - For FAOL , this is the address of a list of longword
003F    ;                  parameters.  For FAO , this is the first of a
003F    ;                  variable number of parameters which
003F    ;                  may have been passed on the call argument list.
003F    ;
003F    ; IMPLICIT INPUTS:
003F    ;
003F    ;       none
003F    ;
003F    ; OUTPUT PARAMETERS:
003F    ;
003F    ;       OUTLEN  - Word pointed to will receive length of output buffer.
003F    ;
003F    ; IMPLICIT OUTPUTS:
003F    ;
003F    ;       none
003F    ;
003F    ; COMPLETION CODES:
003F    ;
003F    ;       SS$_NORMAL      - success code, normal return
003F    ;       SS$_BUFFEROVF   - output buffer overflow, attempt to write past end of outpu
003F    ;       SS$_BADPARAM    - invalid directive specified
003F    ;       SS$_ACCVIO      - unable to read argument list or address arguments
003F    ;
003F    ; SIDE EFFECTS:
003F    ;
003F    ;       none
003F    ;
003F    ;--
003F
003F    ;
003F    ; Global register usage:
003F    ;
003F    ;       R7,R8 - scratch registers
003F    ;       R9  - number of characters remaining in output buffer
```

```
                                003F  ;       R10 - current position in output buffer
                                003F  ;       R11 - pointer to next conversion parameter
                                003F  ;
                                003F  ; Locals
                                003F  ;
                                003F  ;       INLEN(FP) - (word) length of input control string
                                003F  ;       INPTR(FP) - address of position in input control string
                                003F  ;
                                003F
                                003F  ;
                                003F  ; Entry point for call with multiple arguments on stack
                                003F  ;
                                003F
                                003F  EXE$FAO::
                                003F
                          0FFC  003F          .WORD   ^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11>    ; save all registers
         6D   05AD'CF      9E   0041          MOVAB   W^HANDLER,(FP)              ; Establish condition handler
         5B      10 AC     DE   0046          MOVAL   FIRSTARG(AP),R11            ; get address of first argument
                    0B     11   004A          BRB     FAO                         ; go to main routine
                                004C
                                004C  ;
                                004C  ; Entry point for FAOL call.
                                004C  ;
                                004C
                                004C  EXE$FAOL::
                                004C
                          0FFC  004C          .WORD   ^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11>
         6D   05AD'CF      9E   004E          MOVAB   W^HANDLER,(FP)              ; Establish condition handler
         5B      10 AC     D0   0053          MOVL    FIRSTARG(AP),R11            ; address of first argument
                                0057  FAO:
                    7E     7C   0057          CLRQ    -(SP)                       ; save space for LASTVAL and FIELDEND
         7E      04 BC     7D   0059          MOVQ    @INDSC(AP),-(SP)            ; save locals on stack
         59      0C BC     7D   005D          MOVQ    @OUTDSC(AP),R9              ; load output descriptor into R9,R10
              59    59     3C   0061          MOVZWL  R9,R9                       ; ensure word length
                                0064
                                0064  ;
                                0064  ; Look for a control character in the input string.  Copy text
                                0064  ; up to the control, if any , to the output buffer.
                                0064  ;
                                0064
                                0064  MAIN_SCAN:
                    7E     D4   0064          CLRL    -(SP)                       ; indicate control not found
   F4 BD   F0 AD    21     3A   0066          LOCC    #EXCL,INLEN(FP),@INPTR(FP)  ; search for control char
                    02     13   006C          BEQL    10$                         ; branch if not found
                    6E     D6   006E          INCL    (SP)                        ; set indicator to show char. found
                                0070  10$:
      56   F0 AD    50     A3   0070          SUBW3   R0,INLEN(FP),R6             ; calculate bytes to move
           F0 AD    50     D0   0075          MOVL    R0,INLEN(FP)                ; update input length remaining
              59    56     A2   0079          SUBW    R6,R9                       ; update and test output length
                    73     19   007C          BLSS    OVERFLOW                    ; not enough room, error exit
      6A   F4 BD    56     28   007E          MOVC3   R6,@INPTR(FP),(R10)         ; move text part of input string
                 74 8E     E9   0083          BLBC    (SP)+,DONE                  ; leave if no controls left
           F4 AD    51     D0   0086          MOVL    R1,INPTR(FP)                ; update input address pointer
              5A    53     D0   008A          MOVL    R3,R10                      ; update output address pointer
                    7A     10   008D          BSBB    GETCHAR                     ; skip control char
                                008F
                                008F  ;
```

```
                        008F    ; Parse the directive which has been found in the input string.  Set
                        008F    ; up: R0 = remaining count in CNTRL_TABLE
                        008F    ;       R4 = second char if two-char directive
                        008F    ;       R5 = repeat count
                        008F    ;       R6 = field width
                        008F    ;
                        008F
                        008F    PARSE_DIRECTIVE:
                        008F
         55    01  D0   008F            MOVL    #1,R5                   ; default repeat count is 1
               52  D4   0092            CLRL    R2                      ; paren indicator ( not found yet )
             007F  30   0094            BSBW    GETCOUNT                ; pull off count, if any
               70  10   0097            BSBB    GETCHAR                 ; get next char from input string
         53    28  91   0099            CMPB    #^A/(/,R3               ; was next char a paren?
               0D  12   009C            BNEQ    20$                     ; branch if not
               52  D6   009E            INCL    R2                      ; set paren found indicator
               56  D5   00A0            TSTL    R6                      ; was there a repeat count?
               03  19   00A2            BLSS    10$                     ; no..use default
         55    56  D0   00A4            MOVL    R6,R5                   ; else get repeat count
                        00A7    10$:
               6D  10   00A7            BSBB    GETCOUNT                ; look for field width
               5E  10   00A9            BSBB    GETCHAR                 ; get next char
                        00AB    20$:
FF5F CF  10    53  3A   00AB            LOCC    R3,#CNTRL_LENGTH,CNTRL_TABLE ; check character in table
               39  13   00B1            BEQL    ILLEGAL                 ; illegal directive exit
         08    50  D1   00B3            CMPL    R0,#ONECHAR_INDEX       ; is this a one char directive?
               05  15   00B6            BLEQ    30$                     ; yes, don't need any more
               4F  10   00B8            BSBB    GETCHAR                 ; get second control char
         54    53  D0   00BA            MOVL    R3,R4                   ; move to R4 for return
                        00BD    30$:
            02 52  E9   00BD            BLBC    R2,40$                  ; skip if no paren found
               47  10   00C0            BSBB    GETCHAR                 ; else skip paren char
                        00C2    40$:
                        00C2
      53 10    50  C3   00C2            SUBL3   R0,#CNTRL_LENGTH,R3     ; compute offset for case table
                        00C6
                        00C6    ;
                        00C6    ; The following  does a BSBB to the case dispatch
                        00C6    ; table.  The service routines do an RSB and return into CASE_LOOP.
                        00C6    ;
                        00C6
               02  11   00C6            BRB     CASE_LOOP               ; start processing loop
                        00C8    CASE_BSB:
               05  10   00C8            BSBB    FAO_CASE                ; dispatch next directive
                        00CA    CASE_LOOP:
            FB 55  F4   00CA            SOBGEQ  R5,CASE_BSB             ; repeat as specified
               95  11   00CD            BRB     MAIN_SCAN               ; else continue string processing
                        00CF
                        00CF    ;
                        00CF    ; Here is the main dispatch table for dispatching FAO service
                        00CF    ; routines.  The case is entered via BSBB from CASE_BSB. The routines
                        00CF    ; RSB to CASE_LOOP.  Since the 5 numeric conversion directives all
                        00CF    ; dispatch to the same routine, the case has a base of 5 and the
                        00CF    ; numeric directives fall through to the statement following the CASE.
                        00CF    ;
                        00CF    ; Registers R0, R1, and R2 may be scratched by service routines.
                        00CF    ;
```

```
                          00CF
                          00CF  FAO_CASE:
                          00CF          CASE    R3,<-                       ; dispatch to service routine
                          00CF                  CVTASC,-                    ; ascii string insertion
                          00CF                  PERCENT,-                   ; insert ascii time, plural 'S', or UIC
                          00CF                  REPEATIT,-                  ; repeat character 'n' times
                          00CF                  INCR_ARGPTR,-               ; skip next parameter
                          00CF                  DECR_ARGPTR,-               ; backup to previous parameter
                          00CF                  STARTFIELD,-                ; define fixed length field
                          00CF                  ENDFIELD,-                  ; terminate fixed length field
                          00CF                  NEWLINE,-                   ; insert CR/LF
                          00CF                  INSERT_CHAR,-               ; insert TAB
                          00CF                  INSERT_CHAR,-               ; insert form feed
                          00CF                  INSERT_CHAR,-               ; insert '!'
                          00CF          >,B,#5                              ; offset start by 5
                          00E9
             00E0    31   00E9          BRW     CVTNUM                      ; dispatch to numeric conversion
                          00EC
                          00EC
                          00EC  ILLEGAL:
          50   14    3C   00EC          MOVZWL  #SS$_BADPARAM,R0            ; error return code
               0C    11   00EF          BRB     FAO_EXIT
                          00F1  OVERFLOW:
     50   0601 8F    3C   00F1          MOVZWL  #SS$_BUFFEROVF,R0           ; error return code
               59    D4   00F6          CLRL    R9                          ; ensure correct return length
               03    11   00F8          BRB     FAO_EXIT
                          00FA  DONE:
          50   01    3C   00FA          MOVZWL  #SS$_NORMAL,R0              ; no errors
                          00FD  FAO_EXIT:
            08 AC    D5   00FD          TSTL    OUTLEN(AP)                  ; was a return length required?
               06    13   0100          BEQL    10$                         ; branch if not
08 BC   0C BC  59    A3   0102          SUBW3   R9,@OUTDSC(AP),@OUTLEN(AP) ; compute and return output buffer length
                     04   0108  10$:    RET
```

```
                        0109            .SBTTL  GETCHAR - Routine to get next char from input string
                        0109
                        0109    ;++
                        0109    ;
                        0109    ; FUNCTIONAL DESCRIPTION:
                        0109    ;
                        0109    ;       This routine gets the next character from the input control
                        0109    ;       string, updating the length and address pointers.  If the length
                        0109    ;       goes negative, an error exit is called.
                        0109    ;
                        0109    ; CALLING SEQUENCE:
                        0109    ;
                        0109    ;       JSB (R8)
                        0109    ;
                        0109    ; INPUT PARAMETERS:
                        0109    ;
                        0109    ;       none
                        0109    ;
                        0109    ; IMPLICIT INPUTS:
                        0109    ;
                        0109    ;       INLEN(FP) -  lower word has remaining length of input string
                        0109    ;       INPTR(FP) -  is pointer to current string position
                        0109    ;
                        0109    ; OUTPUTS:
                        0109    ;
                        0109    ;       R3 - next character in input string
                        0109    ;
                        0109    ; IMPLICIT OUTPUTS:
                        0109    ;
                        0109    ;       none
                        0109    ;
                        0109    ; COMPLETION CODES:
                        0109    ;
                        0109    ;       none
                        0109    ;
                        0109    ; SIDE EFFECTS:
                        0109    ;
                        0109    ;       input pointers on stack are updated
                        0109    ;       error may cause jump to ILLEGAL
                        0109    ;--
                        0109
                        0109    GETCHAR:
         F0 AD     B7   0109            DECW    INLEN(FP)               ; decr input length remaining
            DE     19   010C            BLSS    ILLEGAL                 ; error if no more left
    53   F4 BD     9A   010E            MOVZBL  @INPTR(FP),R3           ; get next character
         F4 AD     D6   0112            INCL    INPTR(FP)               ; update pointer
                   05   0115            RSB                             ; return
```

```
                         0116        .SBTTL  GETCOUNT - Routine to get repeat-count or field-width
                         0116
                         0116  ;++
                         0116  ;
                         0116  ; FUNCTIONAL DESCRIPTION:
                         0116  ;
                         0116  ;       This subroutine to PARSE_DIRECTIVE scans for a repeat-count or
                         0116  ;       field-width in the directive in the input stream.  If a numeric
                         0116  ;       count is found, it is converted to binary.  If a '#' character
                         0116  ;       is found, the count is taken from the next parameter
                         0116  ;       in the parameter list.
                         0116  ;
                         0116  ; CALLING SEQUENCE:
                         0116  ;
                         0116  ;       JSB or BSB
                         0116  ;
                         0116  ; INPUTS:
                         0116  ;
                         0116  ;       R11     - parameter pointer
                         0116  ;
                         0116  ; IMPLICIT INPUTS:
                         0116  ;
                         0116  ;       none
                         0116  ;
                         0116  ; OUTPUTS:
                         0116  ;
                         0116  ;       R6      - value of count, if # or number found, else -1
                         0116  ;
                         0116  ; IMPLICIT OUTPUTS:
                         0116  ;
                         0116  ;       R11 may be modified if a parameter is taken from the stack
                         0116  ;
                         0116  ; COMPLETION CODES:
                         0116  ;
                         0116  ;       none
                         0116  ;
                         0116  ; SIDE EFFECTS:
                         0116  ;
                         0116  ;       R1, R3, and R4 are destroyed
                         0116  ;--
                         0116
                         0116
                         0116
                         0116  GETCOUNT:
          56    01   CE  0116        MNEGL   #1,R6                   ; not found indicator
       F4 BD    23   91  0119        CMPB    #^A/#/,@INPTR(FP)       ; is this a param. count?
                26   13  011D        BEQL    40$                     ; yes .. pull next param
                53   7C  011F        CLRQ    R3                      ; zero buffer for digit (R3)
                         0121                                        ; ... and accumulator for sum (R4)
     51    F4 AD     D0  0121        MOVL    INPTR(FP),R1            ; remember where we were
                         0125  10$:
  53   F4 BD    30   83  0125        SUBB3   #^A/0/,@INPTR(FP),R3    ; subtract ascii 0 from char
                0F   19  012A        BLSS    20$                     ; branch if not numeric
          53    09   91  012C        CMPB    #^A/9/-^A/0/,R3         ; still numeric?
                0A   19  012F        BLSS    20$                     ; no, branch
          54    0A   C4  0131        MULL2   #10,R4                  ; shift for next digit
          54    53   C0  0134        ADDL    R3,R4                   ; add in next digit
```

```
                D0   10   0137              BSBB    GETCHAR                 ; skip digit we took
                EA   11   0139              BRB     10$                     ; continue while numeric
                          013B         20$:
       F4 AD    51   D1   013B              CMPL    R1,INPTR(FP)            ; did we get any chars?
                03   13   013F              BEQL    30$                     ; no, leave
          56    54   D0   0141              MOVL    R4,R6                   ; yes, return value
                          0144         30$:
                     05   0144              RSB                             ; return
                          0145
                          0145         40$:
          56    8B   D0   0145              MOVL    (R11)+,R6               ; get value from next parameter
                BF   10   0148              BSBB    GETCHAR                 ; skip '#'
                05        014A              RSB                             ; return
```

```
                          014B          .SBTTL  CVTASC  - Insert ASCII string
                          014B
                          014B    ;++
                          014B    ;
                          014B    ; FUNCTIONAL DESCRIPTION:
                          014B    ;
                          014B    ;       Service routine to handle ASCII string insertions.
                          014B    ;       Strings are specified by several different methods.  For
                          014B    ;       filled strings (AF) , non-printing characters are output
                          014B    ;       as dots ('.').
                          014B    ;
                          014B    ; CALLING SEQUENCE:
                          014B    ;
                          014B    ;       JSB or BSB
                          014B    ;
                          014B    ; INPUTS:
                          014B    ;
                          014B    ;       R3      - index of first control char in CNTRL_TABLE
                          014B    ;       R4      - second control character
                          014B    ;       R6      - output field width
                          014B    ;       R9      - output buffer length remaining
                          014B    ;       R10     - output buffer pointer
                          014B    ;       R11     - parameter pointer
                          014B    ;
                          014B    ; IMPLICIT INPUTS:
                          014B    ;
                          014B    ;       none
                          014B    ;
                          014B    ; OUTPUTS:
                          014B    ;
                          014B    ;       none
                          014B    ;
                          014B    ; IMPLICIT OUTPUTS:
                          014B    ;
                          014B    ;       R9 and R10 are update to point to current position in output buffer
                          014B    ;       R11 is updated as parameters are taken from the stack
                          014B    ;
                          014B    ; ROUTINE VALUE:
                          014B    ;
                          014B    ;       none
                          014B    ;
                          014B    ; SIDE EFFECTS:
                          014B    ;
                          014B    ;       R7 and R8 are destroyed
                          014B    ;--
                          014B
                          014B
                          014B    CVTASC:
                          014B
         0078 8F     BB   014B          PUSHR   #^M<R3,R4,R5,R6>        ; save registers
                57   D4   014F          CLRL    R7                      ; set filled indicator to not filled
FEC9 CF   04    54   3A   0151          LOCC    R4,#4,STRING_TYPES      ; search for string subtype
                70   13   0157          BEQL    110$                    ; error if not found
                          0159
                          0159    ;
                          0159    ;       R0 = 1 - filled , 2 - 2 arg desc. , 3 - str. desc. , 4 - cstring
                          0159    ;
```

```
                        0159          CASE    R0,<10$,20$,30$>,B,#2   ; case on descriptor type, base = 2
                        0163
                        0163     ;
                        0163     ; Case falls through here for filled ascii strings. Two argument
                        0163     ; descriptor is used.
                        0163     ;
                        0163
               57   D6  0163          INCL    R7                      ; set filled indicator for filled ascii
                        0165     10$:
          51   8B   7D  0165          MOVQ    (R11)+,R1               ; get length and address
               0E   11  0168          BRB     40$                     ; continue
                        016A
                        016A     ;
                        016A     ; Standard system string descriptor
                        016A     ;
                        016A
                        016A     20$:
          51   9B   7D  016A          MOVQ    @(R11)+,R1              ; move descriptor to R1,R2
          51   51   3C  016D          MOVZWL  R1,R1                   ; make sure length is word
               06   11  0170          BRB     40$                     ; continue
                        0172
                        0172     ;
                        0172     ; Ascii counted string, first byte contains length
                        0172     ;
                        0172
                        0172     30$:
          52   8B   D0  0172          MOVL    (R11)+,R2               ; address of counted string
          51   82   9A  0175          MOVZBL  (R2)+,R1                ; get length and skip byte count
                        0178
                        0178     40$:
                        0178
                        0178     ;
                        0178     ; Here, R1 has string length, R2 has string address. Check length against
                        0178     ; specified field width to decide how much string to move.
                        0178     ;
                        0178
          58   56   D0  0178          MOVL    R6,R8                   ; was a width specified?
               03   18  017B          BGEQ    50$                     ; branch if so
          58   51   D0  017D          MOVL    R1,R8                   ; if not, use string length instead
                        0180     50$:
                        0180
                        0180     ;
                        0180     ; The string is moved to the output buffer with blank fill at the
                        0180     ; end.  The output pointers are then updated by the field width, so
                        0180     ; that the string will be truncated if it was longer than the field
                        0180     ; width.  If the string is filled, a second pass is made to change
                        0180     ; non-printing characters to dots.
                        0180     ;
                        0180
          56   59   D0  0180          MOVL    R9,R6                   ; copy remaining char count
                        0183                                          ; NOTE we have to use R6 here.
          59   58   C2  0183          SUBL    R8,R9                   ; update length remaining
               03   19  0186          BLSS    55$                     ; Overflow, use remaining length.
          56   58   D0  0188          MOVL    R8,R6                   ; else move only required length
6A   56   20   62   51   2C  018B  55$:  MOVC5   R1,(R2),#^A/ /,R6,(R10) ; move string, fill at end
                        0191
          52   5A   D0  0191          MOVL    R10,R2                  ; save output address
```

```
        5A   56   C0 0194         ADDL    R6,R10                  ; update output pointer
          23 57   E9 0197         BLBC    R7,90$                  ; all done if not filled ASCII
                     019A  60$:                                   ; R7 will now become loop counter.
        54   62   9A 019A         MOVZBL  (R2),R4                 ; Fetch character
                     019D  ;
                     019D  ; Check for 7 bit printing (left half of DEC169)
                     019D  ;
        20   54   91 019D         CMPB    R4,#^040                ; Less than space?
             12   1F 01A0         BLSSU   70$                     ; if yes, "."
     7E 8F   54   91 01A2         CMPB    R4,#^0176               ; Less than delete?
             0F   1B 01A6         BLEQU   80$                     ; yes, printing GL
                     01A8  ;
                     01A8  ; Check for 8 bit printing. note that space with 8th bit set is non-printing.
                     01A8  ;
     A0 8F   54   91 01A8         CMPB    R4,#^X80+^040           ; delete or C1 control?
             06   1B 01AC         BLEQU   70$                     ; if yes, "."
     FF 8F   54   91 01AE         CMPB    R4,#^XFF                ; 8 bit "delete" is non-printing
             03   12 01B2         BNEQU   80$                     ; GR printing if not
                     01B4  70$:
        62   2E   90 01B4         MOVB    #^A/./,(R2)             ; Set character to "."
                     01B7  80$:
             52   D6 01B7         INCL    R2                      ; point to next character
     DD 57   56   F3 01B9         AOBLEQ  R6,R7,60$               ; continue until done
                     01BD  90$:
             59   D5 01BD         TSTL    R9                      ; Did we get result overflow above?
             05   19 01BF         BLSS    100$                    ; Yes, branch to tell user.
      0078 8F     BA 01C1         POPR    #^M<R3,R4,R5,R6>
                  05 01C5         RSB                             ; return
                     01C6
                     01C6  100$:
         FF28     31 01C6         BRW     OVERFLOW
                     01C9  110$:
         FF20     31 01C9         BRW     ILLEGAL
```

```
01CC            .SBTTL  CVTNUM - Convert numeric parameter to ASCII
01CC
01CC    ;++
01CC    ;
01CC    ; FUNCTIONAL DESCRIPTION:
01CC    ;
01CC    ;       This routine handles the various HEX, OCTAL, and DECIMAL
01CC    ;       conversions. The proper field is extracted from the
01CC    ;       parameter (byte, word , or long) and the needed output
01CC    ;       width is determined.  This is compared with the user
01CC    ;       specified field width to determine if padding of filling
01CC    ;       is needed.  The entire field with fill is built on the
01CC    ;       stack and then moved so that the result will be correct
01CC    ;       on buffer overflow.
01CC    ;
01CC    ; CALLING SEQUENCE:
01CC    ;
01CC    ;       JSB or BSB
01CC    ;
01CC    ; INPUTS:
01CC    ;
01CC    ;       R3      - index of directive in CNTRL_TABLE.
01CC    ;                       0 = Octal
01CC    ;                       1 = heX
01CC    ;                       2 = Unsigned decimal
01CC    ;                       3 = Signed decimal
01CC    ;                       4 = Zero filled unsigned decimal
01CC    ;       R4      - second char of directive (B,W, or L)
01CC    ;       R6      - field width, or -1 if none
01CC    ;       R9      - output length remaining
01CC    ;       R10     - output position pointer
01CC    ;       R11     - next parameter pointer
01CC    ;
01CC    ; IMPLICIT INPUTS:
01CC    ;
01CC    ;       none
01CC    ;
01CC    ; OUTPUTS:
01CC    ;
01CC    ;       none
01CC    ;
01CC    ; IMPLICIT OUTPUTS:
01CC    ;
01CC    ;       none
01CC    ;
01CC    ; ROUTINE VALUE:
01CC    ;
01CC    ;       none
01CC    ;
01CC    ; SIDE EFFECTS:
01CC    ;
01CC    ;       none
01CC    ;--
01CC
01CC    ;
01CC    ; The registers will be set up as follows
01CC    ;
```

```
                          01CC ;          R0  - max digits to be output
                          01CC ;          R1  - 0 -> byte, 1 -> word , 2 -> long
                          01CC ;          R2  - value to be converted
                          01CC ;          R4  - conversion radix
                          01CC ;          R5  - sign indicator, 1 -> sign to be output, 0 otherwise
                          01CC ;          R7  - fill character, (blank, zero for !Z, or * on width too small
                          01CC ;                for decimal conversions)
                          01CC ;          R8  - total width of field to be output
                          01CC ;
                          01CC
                          01CC CVTNUM:
                          01CC
                   38  BB 01CC          PUSHR   #^M<R3,R4,R5>
                          01CE
      FE50 CF   03 54  3A 01CE          LOCC    R4,#3,DATA_TYPES         ; determine data type
                   03  12 01D4          BNEQ    10$                      ; continue if legal directive
                 FF13  31 01D6          BRW     ILLEGAL                  ; else take error condition
                          01D9 10$:
           51   03 50  C3 01D9          SUBL3   R0,#3,R1                 ; convert to index
                52 8B  D0 01DD          MOVL    (R11)+,R2                ; get next longword parameter
  52   52 FE46 CF41 00  EF 01E0          EXTZV   #0,FIELDS[R1],R2,R2      ; select proper field
                   55  D4 01E8          CLRL    R5                       ; note unsigned
                57 20  90 01EA          MOVB    #^A/ /,R7                ; default fill char is blank
      54  FE48 CF43    9A 01ED          MOVZBL  RADIX[R3],R4             ; get conversion radix
                          01F3
                          01F3 ;
                          01F3 ; Case on the type of conversion.  Note that base is set
                          01F3 ; so that octal and hex conversions fall through case table.
                          01F3 ;
                          01F3
                          01F3          CASE    R3,<40$,30$,20$>,,#2     ; base index of 2
                          01FD
                          01FD ;
                          01FD ; Octal and Hex fall through here
                          01FD ;
                          01FD
           50  6143    DE 01FD          MOVAL   (R1)[R3],R0              ; compute index in OCT_HEX_DIGITS
      50  FE2D CF40    9A 0201          MOVZBL  OCT_HEX_DIGITS[R0],R0    ; get number of digits to output
                58 56  D0 0207          MOVL    R6,R8                    ; user specified width?
                   03  18 020A          BGEQ    15$                      ; yes, use it as width
                58 50  D0 020C          MOVL    R0,R8                    ; else take needed space
                          020F 15$:
                50 58  D1 020F          CMPL    R8,R0                    ; width lss default digits?
                   45  18 0212          BGEQ    60$                      ; no, fill to user specified width
                50 58  D0 0214          MOVL    R8,R0                    ; else output only specified width
                   40  11 0217          BRB     60$
                          0219
                          0219 ;
                          0219 ; Unsigned decimal with zero fill
                          0219 ;
                          0219
                          0219 20$:
                57 30  90 0219          MOVB    #^A/0/,R7                ; insert new fill char
                   11  11 021C          BRB     40$                      ; continue with normal dec. code
                          021E
                          021E ;
                          021E ; Signed decimal conversion
```

```
                                 021E      ;
                                 021E
                                 021E      30$:
 52   52  FE08 CF41    00    EE  021E             EXTV     #0,FIELDS[R1],R2,R2        ; sign extend the field
                05 52  1F    E1  0226             BBC      #31,R2,40$                 ; not negative, continue
                       55    D6  022A             INCL     R5                         ; else note that value negative
                   52  52    CE  022C             MNEGL    R2,R2                      ; and make it positive
                                 022F
                                 022F      40$:                                       ; common decimal processing
                                 022F
                                 022F      ;
                                 022F      ; Determine the number of digits needed to print number in ASCII
                                 022F      ; decimal representation.
                                 022F      ;
                                 022F
                   50  01    D0  022F             MOVL     #1,R0                      ; init digit counter
                   53  54    D0  0232             MOVL     R4,R3                      ; copy first power of 10
                                 0235      44$:
                   53  52    D1  0235             CMPL     R2,R3                      ; does it fit?
                       07    1F  0238             BLSSU    48$                        ; yes, R0 has count if so
                   53  54    C4  023A             MULL     R4,R3                      ; else compute next power of ten
                F4 50  54    F2  023D             AOBLSS   R4,R0,44$                  ; continue (10 digits is largest possible)
                                 0241      48$:
            53     55  50    C1  0241             ADDL3    R0,R5,R3                   ; add in sign, if one exists
                   58  56    D0  0245             MOVL     R6,R8                      ; did user specify width?
                       05    18  0248             BGEQ     50$                        ; yes, use it for field width
                   58  53    D0  024A             MOVL     R3,R8                      ; else use amount needed
                       0A    11  024D             BRB      60$                        ; continue
                                 024F      50$:
                   58  53    D1  024F             CMPL     R3,R8                      ; is there space within specified width?
                       05    15  0252             BLEQ     60$                        ; yes, go on
                   57  2A    90  0254             MOVB     #^A/*/,R7                  ; no room, fill with stars
                       50    D4  0257             CLRL     R0                         ; output no digits
                                 0259
                                 0259      60$:
                F8 AD  52    D0  0259             MOVL     R2,LASTVAL(FP)             ; remember value to be converted
                                 025D
                                 025D      ;
                                 025D      ; Insert the ASCII representation for the value in R2 into the
                                 025D      ; output buffer.
                                 025D      ;
                                 025D
                                 025D      CVT_BIN_TO_ASC:
                                 025D
              0840 8F        BB  025D             PUSHR    #^M<R6,R11>                ; save work registers
                04 A8        9F  0261             PUSHAB   4(R8)                      ; compute stack space needed for buffer
                   6E  03    CA  0264             BICL     #3,(SP)                    ; round stack to longword
                   5B  5E    D0  0267             MOVL     SP,R11                     ; save stack pointer
                   5E  6B    C2  026A             SUBL     (R11),SP                   ; leave buffer space on stack
                                 026D
                       53    D4  026D             CLRL     R3                         ; clear upper half of quad quotient
                   51  01    CE  026F             MNEGL    #1,R1                      ; init digit counter for loop
                       0B    11  0272             BRB      15$                        ; start loop
                                 0274      10$:
    56  52    52   54        7B  0274             EDIV     R4,R2,R2,R6                ; R2 <- quotient, R6 <- remainder
       7B   FD82 CF46        90  0279             MOVB     ASC_NAMES[R6],-(R11)       ; output ascii digit
                                 027F      15$:
```

```
   F1 51  50  F2 027F            AOBLSS  R0,R1,10$               ; one more digit, done yet?
      05  55  E9 0283            BLBC    R5,20$                  ; branch if no sign to output
   7B     2D  90 0286            MOVB    #^A/-/,-(R11)           ; output sign
          51  D6 0289            INCL    R1                      ;
                 028B      20$:
                 028B
                 028B      ;
                 028B      ; If field (R8) is not full, then fill remainder with the fill character
                 028B      ;
                 028B
          03  11 028B            BRB     40$                     ; start the loop
                 028D      30$:
      7B  57  90 028D            MOVB    R7,-(R11)               ; insert fill character
                 0290      40$:
   F9 51  58  F3 0290            AOBLEQ  R8,R1,30$               ; fill until full
                 0294
                 0294      ;
                 0294      ; Now copy stack back to buffer, checking for overflow
                 0294      ;
                 0294
          08  11 0294            BRB     70$                     ; start loop
                 0296      50$:
      02  59  F4 0296            SOBGEQ  R9,60$                  ; update length, check for overflow
          21  11 0299            BRB     INSERT_OVF              ; handle overflow
      8A  8B  90 029B      60$:  MOVB    (R11)+,(R10)+           ; move char to output buffer
      F5  58  F4 029E      70$:  SOBGEQ  R8,50$                  ; move entire string
                 02A1
                 02A1      ;
                 02A1      ; Now clean up mess on stack
                 02A1      ;
                 02A1
      5E  5B  D0 02A1            MOVL    R11,SP                  ; restore stack
    0841  8F  BA 02A4            POPR    #^M<R0,R6,R11>          ; remove top of stack and restore regs
                 02A8
                 02A8      ;
                 02A8      ; Restore registers and return from service routine.
                 02A8      ;
                 02A8
          38  BA 02A8            POPR    #^M<R3,R4,R5>
              05 02AA            RSB
              02AB          .SBTTL  QUICKSERVE - Small service routines
              02AB
              02AB    ;++
              02AB    ;
              02AB    ; FUNCTIONAL DESCRIPTION:
              02AB    ;
              02AB    ;       Following are a collection of short service routines for
              02AB    ;       FAO directives.
              02AB    ;
              02AB    ; CALLING SEQUENCE:
              02AB    ;
              02AB    ;       JSB or BSB
              02AB    ;
              02AB    ; INPUTS:
              02AB    ;
              02AB    ;       R3  - index in CNTRL_TABLE of the directive
              02AB    ;       R4  - second character of two-char directive, if any
              02AB    ;       R6  - user specified field width, if any (ignored for singal char
              02AB    ;             and argument directives)
              02AB    ;       R9  - output length remaining
              02AB    ;       R10 - output position pointer
              02AB    ;
              02AB    ; IMPLICIT INPUTS:
              02AB    ;
              02AB    ;       none
              02AB    ;
              02AB    ; OUTPUTS:
              02AB    ;
              02AB    ;       none
              02AB    ;
              02AB    ; IMPLICIT OUTPUTS:
              02AB    ;
              02AB    ;       R9 and R10 are modified
              02AB    ;
              02AB    ; COMPLETION CODES:
              02AB    ;
              02AB    ;       none
              02AB    ;
              02AB    ; SIDE EFFECTS:
              02AB    ;
              02AB    ;       none
              02AB    ;--
              02AB
              02AB
              02AB    INCR_ARGPTR:
              02AB
              02AB    ;
              02AB    ; Directive to skip next parameter in parameter list
              02AB    ;
              02AB
     8B  D5   02AB          TSTL    (R11)+                  ; skip next parameter
         05   02AD          RSB                             ; exit
              02AE
              02AE    DECR_ARGPTR:
              02AE
              02AE    ;
              02AE    ; Directive to back up and reuse last parameter in parameter list
```

```
                                02AE    ;
                                02AE
                        7B  D5  02AE            TSTL    -(R11)                  ; back up argument pointer
                            05  02B0            RSB                             ; exit
                                02B1
                                02B1    NEWLINE:
                                02B1
                                02B1    ;
                                02B1    ; Insert carriage return, line feed into output buffer
                                02B1    ;
                                02B1
                     02 59  F4  02B1            SOBGEQ  R9,10$                  ; room for CR?, branch if so
                        06  11  02B4            BRB     INSERT_OVF              ; no room in output buffer
                                02B6    10$:
                 8A     0D  90  02B6            MOVB    #CR,(R10)+              ; insert CR in output buffer
                                02B9                                            ; continue for LF insertion
                                02B9
                                02B9    INSERT_CHAR:
                                02B9
                                02B9    ;
                                02B9    ; Make simple one character insertion in the output buffer.
                                02B9    ;
                                02B9
                     03 59  F4  02B9            SOBGEQ  R9,INSERT_IT            ; check length, branch if ok
                                02BC    INSERT_OVF:
                      FE32  31  02BC            BRW     OVERFLOW                ; error , no room in output buffer
                                02BF    INSERT_IT:
                                02BF
                                02BF    ;
                                02BF    ; Insert the character by computing the index into the replacement table
                                02BF    ;
                                02BF
           8A   FD5F CF43   90  02BF            MOVB    REPLACEMENT-REPL_OFFSET[R3],(R10)+ ; insert the char
                            05  02C5            RSB
                                02C6
                                02C6    ;
                                02C6    ; Directive to repeat a particular character 'n' times, where 'n' is
                                02C6    ; specified by the field width in the directive.
                                02C6    ;
                                02C6
                                02C6    REPEATIT:
                        38  BB  02C6            PUSHR   #^M<R3,R4,R5>           ; save regs for MOVC5 clobber
                        56  D5  02C8            TSTL    R6                      ; check if width was specified
                        15  19  02CA            BLSS    ILLFIELD                ; illegal if none specified
                     59 56  C2  02CC            SUBL    R6,R9                   ; compute remaining output length
                        EB  19  02CF            BLSS    INSERT_OVF              ; not enough room, error
      6A    56    54  6E 00 2C  02D1            MOVC5   #0,(SP),R4,R6,(R10)     ; fill with specified character
                     5A 56  C0  02D7            ADDL    R6,R10                  ; update output pointer
                        38  BA  02DA            POPR    #^M<R3,R4,R5>           ; restore regs
                            05  02DC            RSB
                                02DD
                                02DD    ;
                                02DD    ; The following are the directives which define a fixed length field.
                                02DD    ; The field width is specified with the define field directive.  At the
                                02DD    ; end field directive, any of the field remaining is blank filled, else
                                02DD    ; the field is truncated to the specified length.
                                02DD    ;
```

```
                          02DD
                          02DD  STARTFIELD:
                  56  D5  02DD             TSTL    R6                      ; did user specify field (must be specified)
                  03  18  02DF             BGEQ    STARTOK                 ; yes, continue
                          02E1  ILLFIELD:
                FE08  31  02E1             BRW     ILLEGAL                 ; illegal directive
     FC AD    5A  56  C1  02E4  STARTOK:ADDL3   R6,R10,FIELDEND(FP)     ; compute and save ending address
              59  56  D1  02E9             CMPL    R6,R9                   ; was that much space remaining?
                  CE  14  02EC             BGTR    INSERT_OVF              ; no, take error here
                      05  02EE             RSB                             ; return
                          02EF
                          02EF  ;
                          02EF  ; Set up registers so that if fill is needed, a phony call is made
                          02EF  ; to REPEATIT with the length in R6 and the 'blank' character in R4
                          02EF  ;
                          02EF
                          02EF  ENDFIELD:
              54  20  9A  02EF             MOVZBL  #^A/ /,R4               ; generate blank fill character
     56    FC AD  5A  C3  02F2             SUBL3   R10,FIELDEND(FP),R6     ; compute remaining field length
                  CD  14  02F7             BGTR    REPEATIT                ; if any left, go fill with blanks
           5A  FC AD  D0  02F9             MOVL    FIELDEND(FP),R10        ; else truncate by setting back pointer
              59  56  C2  02FD             SUBL    R6,R9                   ; subtract negative difference from counter
                  05  0300                 RSB                             ; return
```

```
                                      0301              .SBTTL  PERCENT - Time directives, plural 'S', and UIC
                                      0301
                                      0301       ;++
                                      0301       ;
                                      0301       ; FUNCTIONAL DESCRIPTION:
                                      0301       ;
                                      0301       ;      These directives are for date and time conversion, for
                                      0301       ;      conditionally inserting a plural 'S' into messages, and UIC conversion.
                                      0301       ;      The time directives insert an ASCII time string into the output buffer.
                                      0301       ;      The user may supply a quadword binary time to be converted,
                                      0301       ;      or have the current date or time inserted.
                                      0301       ;
                                      0301       ; CALLING SEQUENCE:
                                      0301       ;
                                      0301       ;      JSB/BSB
                                      0301       ;
                                      0301       ; INPUTS:
                                      0301       ;
                                      0301       ;      R4  - second character of directive.  D -> convert
                                      0301       ;              date and time, T -> convert time only
                                      0301       ;              S -> plural indicator, U -> convert UIC
                                      0301       ;              I -> identifier
                                      0301       ;      R6  - user specified field width, if any
                                      0301       ;      R9  - remaining length of output buffer
                                      0301       ;      R10 - current output buffer position
                                      0301       ;      R11 - next parameter address
                                      0301       ;
                                      0301       ; IMPLICIT INPUTS:
                                      0301       ;
                                      0301       ;      none
                                      0301       ;
                                      0301       ; OUTPUTS:
                                      0301       ;
                                      0301       ;      none
                                      0301       ;
                                      0301       ; IMPLICIT OUTPUTS:
                                      0301       ;
                                      0301       ;      none
                                      0301       ;
                                      0301       ; ROUTINE VALUE:
                                      0301       ;
                                      0301       ;      none
                                      0301       ;
                                      0301       ; SIDE EFFECTS:
                                      0301       ;
                                      0301       ;      none
                                      0301       ;--
                                      0301
                            000001FC  0301              ID_REG_MASK=    ^M<R2,R3,R4,R5,R6,R7,R8>        ; %I & %U WORK REG MASK
                                      0301
                                      0301       PERCENT:
          FD20 CF    05    54    3A   0301              LOCC    R4,#5,PERCENT_STR       ; find directive type
                           D8    13   0307              BEQL    ILLFIELD                ; illegal directive if not found
                           57    D4   0309              CLRL    R7                      ; assume date and time
                                      030B              CASE    R0,<5$,10$,30$,70$,210$>,B,#1   ; branch on directive type
                                      0319
                                      0319       ;
```

```
                                 0319  ; Time only directive falls through here
                                 0319  ;
                                 0319
                                 0319  5$:
                       57    D6  0319          INCL    R7                       ; indicate time only
                                 031B  10$:                                     ; time and date enters here
                       38    BB  031B          PUSHR   #^M<R3,R4,R5>            ; save registers
 6A   59   20   6A     00    2C  031D          MOVC5   #0,(R10),#^A/ /,R9,(R10) ; blank fill rest of output buffer
                58     7E    DE  0323          MOVAL   -(SP),R8                 ; space for return length
                7E     59    7D  0326          MOVQ    R9,-(SP)                 ; form descriptor for output buffer
                52     6E    DE  0329          MOVAL   (SP),R2                  ; get address of buffer descriptor
                51     8B    D0  032C          MOVL    (R11)+,R1                ; get binary time address
                       04    13  032F          BEQL    12$                      ; branch if no address
           D4 A1       61    D1  0331          CMPL    (R1),4(R1)               ; let potential access violation
                                 0335                                           ; ...happen in this frame rather than
                                 0335                                           ; ...within $ASCTIM to help condition
                                 0335                                           ; ...handler
                                 0335  12$:    $ASCTIM_S (R8),(R2),(R1),R7      ; convert time to ascii
                52     56    D0  0344          MOVL    R6,R2                    ; did user specify width?
                       03    18  0347          BGEQ    20$                      ; yes, use it
                52     68    3C  0349          MOVZWL  (R8),R2                  ; else use returned length
                                 034C  20$:
                59     52    C2  034C          SUBL    R2,R9                    ; update output length
                       12    19  034F          BLSS    40$                      ; error, not enough room
                5A     52    C0  0351          ADDL    R2,R10                   ; update output buffer
                5E     0C    C0  0354          ADDL    #12,SP                   ; pop locals from stack
                       38    BA  0357          POPR    #^M<R3,R4,R5>            ; restore registers
                             05  0359          RSB                              ;
                                 035A  30$:
                                 035A
                                 035A  ;
                                 035A  ; Check if the last value converted was equal to one.  If so, then do
                                 035A  ; nothing, else output an 'S' into the output buffer.
                                 035A  ;
                                 035A
           F8 AD       01    D1  035A          CMPL    #1,LASTVAL(FP)           ; was last value a one
                       13    13  035E          BEQL    60$                      ; yes, simply return
                    03 59    F4  0360          SOBGEQ  R9,50$                   ; check if room in buffer
                     FD8B    31  0363  40$:    BRW     OVERFLOW                 ; no room , error
           8A       53 8F    90  0366  50$:    MOVB    #^A/S/,(R10)+            ; plural, insert 'S'
      04 FE AA         05    E1  036A          BBC     #5,-2(R10),60$           ; continue if previous character was
                                 036F                                           ; ...upper case
           FF AA       20    88  036F          BISB    #^X20,-1(R10)            ; else convert upper 'S' to lower 's'
                             05  0373  60$:    RSB                              ; return
                                 0374
                                 0374  ;
                                 0374  ; Convert a longword value to an identifier if possible.  This identifier may
                                 0374  ; take one of two forms, a random identifier or an alphanumeric UIC.  In the
                                 0374  ; case of an alphanumeric UIC, an attempt is first made to translate just the
                                 0374  ; group portion of the UIC.  If this fails, an attempt is made to translate
                                 0374  ; the entire UIC.  If this also fails, the UIC is formatted using the %U
                                 0374  ; directive.
                                 0374
            01FC 8F          BB  0374  70$:    PUSHR   #ID_REG_MASK             ; SAVE WORK REGS
                5E     20    C2  0378          SUBL2   #32,SP                   ; GROUP IDENTIFIER STORAGE
                7E     5E    D0  037B          MOVL    SP,-(SP)                 ; GROUP IDENTIFIER
                       20    DD  037E          PUSHL   #32                      ;   DESCRIPTOR
```

```
          57    5E   D0  0380  1142          MOVL    SP,R7                    ; SAVE DESCRIPTOR ADDRESS FOR LATER
          5E    20   C2  0383  1143          SUBL2   #32,SP                   ; USER IDENTIFIER STORAGE
          7E    5E   D0  0386  1144          MOVL    SP,-(SP)                 ; USER IDENTIFIER
                20   DD  0389  1145          PUSHL   #32                      ;   DESCRIPTOR
          58    5E   D0  038B  1146          MOVL    SP,R8                    ; SAVE DESCRIPTOR ADDRESS FOR LATER
                         038E  1147
                         038E  1148          ASSUME  UIC$K_UIC_FORMAT EQ 0
                         038E  1149          ASSUME  UIC$K_ID_FORMAT EQ 2
                         038E  1150          ASSUME  UIC$V_FORMAT EQ 30
                         038E  1151
          52    6B   D0  038E  1152          MOVL    (R11),R2                 ; GET THE IDENTIFIER NUMBER
                27   19  0391  1153          BLSS    75$                      ; XFER IF NOT A UIC
          52    01   AE  0393  1154          MNEGW   #1,R2                    ; SET UP FOR GROUP IDENTIFIER CHECK
                         0396  1155          $IDTOASC_S      ID=R2,-          ; TRANSLATE TO GROUP NAME IF POSSIBLE
                         0396  1156                          NAMLEN=(R7),-
                         0396  1157                          NAMBUF=(R7)
             0E 50   E9  03A9  1158          BLBC    R0,75$                   ; XFER IF ERRORS IN TRANSLATING
                         03AC  1159
                         03AC  1160          ASSUME  UIC$K_WILD_MEMBER EQ <^XFFFF>
                         03AC  1161
          52    6B   B1  03AC  1162          CMPW    (R11),R2                 ; WILD MEMBER (R2 SET ABOVE)
                0B   12  03AF  1163          BNEQ    80$                      ; XFER IF NOT
          68    01   D0  03B1  1164          MOVL    #1,(R8)                  ; ELSE SET SIZE
       04 B8    2A   90  03B4  1165          MOVB    #^A\*\,@4(R8)            ; SET WILDCARD CHARACTER
                1A   11  03B8  1166          BRB     90$                      ; GO BUILD UIC
                67   B4  03BA  1167  75$:    CLRW    (R7)                     ; ELSE SET FOR ZERO SIZE
                         03BC  1168  80$:    $IDTOASC_S      ID=(R11),-       ; TRANSLATE TO USER NAME IF POSSIBLE
                         03BC  1169                          NAMLEN=(R8),-
                         03BC  1170                          NAMBUF=(R8)
             02 50   E8  03CF  1171          BLBS    R0,90$                   ; XFER IF NO ERRORS
                68   B4  03D2  1172          CLRW    (R8)                     ; ELSE SET ZERO SIZE
          53    67   3C  03D4  1173  90$:    MOVZWL  (R7),R3                  ; GET GROUP NAME SIZE
                02   13  03D7  1174          BEQL    100$                     ; XFER IF GROUP DIDN'T TRANSLATE
                53   D6  03D9  1175          INCL    R3                       ; ELSE ACCOUNT FOR COMMA SEPARATOR
          50    68   3C  03DB  1176  100$:   MOVZWL  (R8),R0                  ; GET USER NAME SIZE
                66   13  03DE  1177          BEQL    150$                     ; XFER IF DIDN'T TRANSLATE
          53    50   C0  03E0  1178          ADDL2   R0,R3                    ; ELSE TOTAL UP THE SIZE
                61   13  03E3  1179          BEQL    150$                     ; XFER IF UIC DIDN'T TRANSLATE
                         03E5  1180
                         03E5  1181          ASSUME  UIC$K_ID_FORMAT EQ 2
                         03E5  1182          ASSUME  UIC$V_FORMAT EQ 30
                         03E5  1183
       03 6B    1F   E0  03E5  1184          BBS     #31,(R11),105$           ; XFER IF NOT UIC
          53    02   C0  03E9  1185          ADDL2   #2,R3                    ; ELSE ACCOUNT FOR SQUARE BRACKETS
                56   D5  03EC  1186  105$:   TSTL    R6                       ; WIDTH SUPPLIED?
                03   18  03EE  1187          BGEQ    110$                     ; XFER IF SO
          56    53   D0  03F0  1188          MOVL    R3,R6                    ; ELSE SET IT
          56    53   D1  03F3  1189  110$:   CMPL    R3,R6                    ; FIELD WIDTH EXCEEDED?
                34   14  03F6  1190          BGTR    130$                     ; XFER IF SO...NOTE IT
          59    53   D1  03F8  1191          CMPL    R3,R9                    ; BUFFER EXCEEDED?
                3E   14  03FB  1192          BGTR    140$                     ; XFER IF SO...NOTE IT
          56    53   D0  03FD  1193          MOVL    R3,R6                    ; ELSE PUTE LENGTH IN NONVOLATILE REG
          53    5A   D0  0400  1194          MOVL    R10,R3                   ; GET OUTPUT BUFFER ADDRESS
                         0403  1195
                         0403  1196          ASSUME  UIC$K_ID_FORMAT EQ 2
                         0403  1197          ASSUME  UIC$V_FORMAT EQ 30
                         0403  1198
```

```
        10 6B    1F  E0 0403          BBS     #31,(R11),120$              ; XFER IF NOT UIC
        83    5B 8F  90 0407          MOVB    #^A/[/,(R3)+                ; NOTE START OF UIC
                 67  B5 040B          TSTW    (R7)                        ; DID GROUP NAME TRANSLATE?
                 08  13 040D          BEQL    120$                        ; XFER IF NOT
   63     04 B7  67  28 040F          MOVC3   (R7),@4(R7),(R3)            ; ELSE COPY GROUP NAME
                 83  2C  90 0414      MOVB    #^A/,/,(R3)+                ; SAVE SEPARATOR
   63     04 B8  68  28 0417  120$:   MOVC3   (R8),@4(R8),(R3)            ; COPY USER NAME
          04 6B  1F  E0 041C          BBS     #31,(R11),125$              ; XFER IF NOT UIC
        83    5D 8F  90 0420          MOVB    #^A/]/,(R3)+                ; TIE OFF THE UIC
              59 56  C2 0424  125$:   SUBL2   R6,R9                       ; CALC REMAINING ROOM IN THE BUFFER
              5A 56  C0 0427          ADDL2   R6,R10                      ; CALC NEXT AVAILABLE IN BUFFER
                 5D  11 042A          BRB     190$                        ; GO FINISH UP
 6A   56  2A     6E  00  2C 042C  130$:   MOVC5   #0,(SP),#^A/*/,R6,(R10) ; NOTE FIELD OVERFLOWED
              59 56  C2 0432          SUBL2   R6,R9                       ; ADJUST COUNT
              5A 56  C0 0435          ADDL2   R6,R10                      ; AND POINTER
               FCB6  31 0438          BRW     OVERFLOW                    ; NOTE OVERFLOW
 6A   59  2A     6E  00  2C 043B  140$:   MOVC5   #0,(SP),#^A/*/,R9,(R10) ; NOTE BUFFER OVERFLOWED
                 59  D4 0441          CLRL    R9                          ; NO ROOM LEFT
               FCAB  31 0443          BRW     OVERFLOW                    ; NOTE OVERFLOW
                        0446
                        0446  ; At this point, it has been determined that no translation exists for the
                        0446  ; specified identifier.  If it is a UIC, format it using the %U.  If it is
                        0446  ; a random identifier, try to convert it to a hex number.
                        0446
     FFFFFFFF 8F     6B  D1 0446  150$:   CMPL    (R11),#-1           ; IS THIS THE MATCH-ALL IDENTIFIER?
                 0B  12 044D          BNEQ    170$                        ; XFER IF NOT
              03 59  F4 044F          SOBGEQ  R9,160$                     ; XFER IF ROOM
               FC9C  31 0452          BRW     OVERFLOW                    ; ELSE NOTE OVERFLOW
              8A 2A  90 0455  160$:   MOVB    #^A\*\,(R10)+               ; NOTE THE MATCH-ALL IDENTIFIER
                 2F  11 0458          BRB     190$                        ; GO FINISH WITH THIS DIRECTIVE
     00   6B   02    1E  ED 045A  170$:   CMPZV   #UIC$V_FORMAT,#UIC$S_FORMAT,(R11),#UIC$K_UIC_FORMAT      ; UIC?
                 36  13 045F          BEQL    200$                        ; XFER IF SO
   5E   00000050 8F  C0 0461          ADDL2   #8+32+8+32,SP               ; CLEAN UP THE STACK
            01FC 8F  BA 0468          POPR    #ID_REG_MASK                ; RESTORE WORK REGS
              03 59  F4 046C          SOBGEQ  R9,180$                     ; INSURE ROOM FOR %X
               FC7F  31 046F          BRW     OVERFLOW
              8A 25  90 0472  180$:   MOVB    #^A\%\,(R10)+
              03 59  F4 0475          SOBGEQ  R9,185$
               FC76  31 0478          BRW     OVERFLOW
           8A 58 8F  90 047B  185$:   MOVB    #^A\X\,(R10)+
              53 01  D0 047F          MOVL    #1,R3                       ; SET UP FOR HEX CONVERSION
           54 4C 8F  9A 0482          MOVZBL  #^A/L/,R4
               FD43  31 0486          BRW     CVTNUM                      ; GO TRY TO CONVERT
   5E   00000050 8F  C0 0489  190$:   ADDL2   #8+32+8+32,SP               ; CLEAN UP THE STACK
            01FC 8F  BA 0490          POPR    #ID_REG_MASK                ; RESTORE WORK REGS
                 8B  D5 0494          TSTL    (R11)+                      ; SET TO NEXT PARAMETER
                     05 0496          RSB                                 ; RETURN FOR MORE
   5E   00000050 8F  C0 0497  200$:   ADDL2   #8+32+8+32,SP               ; CLEAN UP THE STACK
            01FC 8F  BA 049E          POPR    #ID_REG_MASK                ; RESTORE WORK REGISTERS
                        04A2
                        04A2  ;
                        04A2  ; Convert the longword value to a UIC in a standard format.  This format is
                        04A2  ; [group,member].  Where the group and member portions are a word (16-bits)
                        04A2  ; each.  If a width is supplied, the UIC is centered (by the comma) in the
                        04A2  ; field.
                        04A2  ;
                        04A2
```

```
                   01FC 8F   BB  04A2  210$:   PUSHR   #ID_REG_MASK                    ; SAVE WORK REGISTERS
                     5E 10   C2  04A6          SUBL2   #16,SP                          ; MAKE ROOM FOR GROUP & MEMBER
                     57 5E   D0  04A9          MOVL    SP,R7                           ; SET ADDRESS FOR GROUP
               58    08 AE   9E  04AC          MOVAB   8(SP),R8                        ; SET ADDRESS FOR MEMBER
                     50 58   D0  04B0          MOVL    R8,R0                           ; SET ADDRESS OF MEMBER STRING
                        80   B4  04B3          CLRW    (R0)+                           ; RESET CHARACTER COUNT
                     52 0F   D0  04B5          MOVL    #15,R2                          ; SET STARTING BIT
      51      6B     10 00   EF  04B8          EXTZV   #UIC$V_MEMBER,#UIC$S_MEMBER,(R11),R1     ; GET MEMBER NUMBER
            FFFF     8F 51   B1  04BD          CMPW    R1,#UIC$K_WILD_MEMBER           ; IS IT A WILDCARD MEMBER?
                        09   12  04C2          BNEQ    220$                            ; XFER IF NOT
                     68 01   B0  04C4          MOVW    #1,(R8)                         ; ELSE SET SIZE
                     80 2A   90  04C7          MOVB    #^A/*/,(R0)+                    ; SET WILDCARD STRING
                      001F   31  04CA          BRW     250$                            ; GO GET THE GROUP
      53      51     03 52   EF  04CD  220$:   EXTZV   R2,#3,R1,R3                     ; GET AN OCTAL DIGIT
                        04   12  04D2          BNEQ    230$                            ; XFER IF NON-ZERO
                        68   B5  04D4          TSTW    (R8)                            ; ELSE CHECK FOR ZERO SUPPRESSION
                        06   13  04D6          BEQL    240$                            ; XFER IF SUPPRESSING
              80      53 30  81  04D8  230$:   ADDB3   #^A/0/,R3,(R0)+                 ; CONVERT TO ASCII AND SAVE IT
                        68   B6  04DC          INCW    (R8)                            ; ONE MORE CHARACTER
                     52 03   C2  04DE  240$:   SUBL2   #3,R2                           ; SET FOR THE NEXT DIGIT
                        EA   18  04E1          BGEQ    220$                            ; CONTINUE TILL ALL DONE
                        68   B5  04E3          TSTW    (R8)                            ; ANYTHING THERE?
                        05   12  04E5          BNEQ    250$                            ; XFER IF SO
                     80 30   90  04E7          MOVB    #^A/0/,(R0)+                    ; ELSE SAVE AT LEAST ONE ZERO
                        68   B6  04EA          INCW    (R8)                            ; COUNT IT
                     50 57   D0  04EC  250$:   MOVL    R7,R0                           ; SET ADDRESS OF GROUP STRING
                        80   B4  04EF          CLRW    (R0)+                           ; RESET CHARACTER COUNT
                     52 0F   D0  04F1          MOVL    #15,R2                          ; SET STARTING BIT
      51      6B     0E 10   EF  04F4          EXTZV   #UIC$V_GROUP,#UIC$S_GROUP,(R11),R1       ; GET GROUP NUMBER
            3FFF     8F 51   B1  04F9          CMPW    R1,#UIC$K_WILD_GROUP            ; IS IT A WILDCARD GROUP?
                        09   12  04FE          BNEQ    260$                            ; XFER IF NOT
                     67 01   B0  0500          MOVW    #1,(R7)                         ; ELSE SET SIZE
                     80 2A   90  0503          MOVB    #^A\*\,(R0)+                    ; SET WILDCARD STRING
                      001F   31  0506          BRW     290$                            ; GO GET THE GROUP
      53      51     03 52   EF  0509  260$:   EXTZV   R2,#3,R1,R3                     ; GET AN OCTAL DIGIT
                        04   12  050E          BNEQ    270$                            ; XFER IF NON-ZERO
                        67   B5  0510          TSTW    (R7)                            ; ELSE CHECK FOR ZERO SUPPRESSION
                        06   13  0512          BEQL    280$                            ; XFER IF SUPPRESSING
              80      53 30  81  0514  270$:   ADDB3   #^A/0/,R3,(R0)+                 ; CONVERT TO ASCII AND SAVE
                        67   B6  0518          INCW    (R7)                            ; COUNT THE CHARACTER
                     52 03   C2  051A  280$:   SUBL2   #3,R2                           ; SET FOR THE NEXT DIGIT
                        EA   18  051D          BGEQ    260$                            ; CONTINUE TILL DONE
                        67   B5  051F          TSTW    (R7)                            ; ANYTHING THERE?
                        05   12  0521          BNEQ    290$                            ; XFER IF SO
                     80 30   90  0523          MOVB    #^A/0/,(R0)+                    ; ELSE SAVE AT LEAST ONE ZERO
                        67   B6  0526          INCW    (R7)                            ; COUNT IT
                        8B   D5  0528  290$:   TSTL    (R11)+                          ; STEP OVER UIC
                     50 68   B0  052A          MOVW    (R8),R0                         ; GET SIZE OF MEMBER FIELD
                     50 67   A0  052D          ADDW2   (R7),R0                         ; AND GROUP FIELD
                     50 03   A0  0530          ADDW2   #3,R0                           ; PLUS DELIMITERS
                     50 50   3C  0533          MOVZWL  R0,R0                           ; FULL LONGWORD
                        56   D5  0536          TSTL    R6                              ; ANY FIELD WIDTH GIVEN?
                        16   18  0538          BGEQ    300$                            ; XFER IF SO
                     53 5A   D0  053A          MOVL    R10,R3                          ; COPY ADDRESS OF OUTPUT FIELD
                     56 50   D0  053D          MOVL    R0,R6                           ; SET FIELD WIDTH
                     59 50   D1  0540          CMPL    R0,R9                           ; ELSE SEE IF THERE IS ROOM FOR THE UIC
                        36   15  0543          BLEQ    320$                            ; XFER IF THERE IS ROOM
```

```
 6A   59   2A   6E   00  2C  0545          MOVC5   #0,(SP),#^A/*/,R9,(R10) ; ELSE FILL REMAINING ROOM
                     59  D4  054B          CLRL    R9                      ; NO REMANING ROOM
                   FBA1  31  054D          BRW     OVERFLOW                ; ELSE INDICATE THE OVERFLOW
                59   56  D1  0550  300$:   CMPL    R6,R9                   ; IS THERE ROOM IN THE OUTPUT BUFFER?
                     03  15  0553          BLEQ    310$                    ; XFER IF SO
                   0046  31  0555          BRW     330$                    ; ELSE INDICATE OVERFLOW
                56   50  D1  0558  310$:   CMPL    R0,R6                   ; IS THE ROOM IN THE FIELD FOR THE UIC?
                     41  14  055B          BGTR    330$                    ; XFER IF NOT...FIELD WIDTH OVERFLOW
 6A   56   20   6E   00  2C  055D          MOVC5   #0,(SP),#^A/ /,R6,(R10) ; ELSE FILL FIELD FIRST
           51   56   03  C3  0563          SUBL3   #3,R6,R1                ; CALC SIZE MINUS DELIMITERS
                51   02  C6  0567          DIVL2   #2,R1                   ; MAX SIZE FOR CENTERING
                51   67  B1  056A          CMPW    (R7),R1                 ; ROOM FOR GROUP SUBFIELD?
                     2F  14  056D          BGTR    330$                    ; XFER IF NOT...F.W.O.
                51   68  B1  056F          CMPW    (R8),R1                 ; ROOM FOR MEMBER SUBFIELD?
                     2A  14  0572          BGTR    330$                    ; XFER IF NOT...F.W.O.
                51   67  A2  0574          SUBW2   (R7),R1                 ; CALC NUMBER OF LEADING SPACES
           53   5A   51  C1  0577          ADDL3   R1,R10,R3               ; ADJUST FOR LEADING SPACES
           83   5B   8F  90  057B  320$:   MOVB    #^A/[/,(R3)+            ; FIRST DELIMITER
      63   02   A7   67  28  057F          MOVC3   (R7),2(R7),(R3)         ; COPY GROUP SUBFIELD
                83   2C  90  0584          MOVB    #^A/,/,(R3)+            ; SUBFIELD DELIMITER
      63   02   A8   68  28  0587          MOVC3   (R8),2(R8),(R3)         ; COPY MEMBER SUBFIELD
           83   5D   8F  90  058C          MOVB    #^A/]/,(R3)+            ; TIE OFF THE UIC
                5A   56  C0  0590          ADDL2   R6,R10                  ; CALC NEXT AVAILABLE POSITION
                59   56  C2  0593          SUBL2   R6,R9                   ; CALC REMAINING BUFFER POSITIONS
                5E   10  C0  0596          ADDL2   #16,SP                  ; CLEAN UP THE STACK
              01FC   8F  BA  0599          POPR    #ID_REG_MASK            ; RESTORE WORK REGISTERS
                         05  059D          RSB                             ; AND RETURN
 6A   56   2A   6E   00  2C  059E  330$:   MOVC5   #0,(SP),#^A/*/,R6,(R10) ; INDICATE OVERFLOW
                5A   56  C0  05A4          ADDL2   R6,R10                  ; POINT TO NEXT FIELD IN OUTPUT
                59   56  C2  05A7          SUBL2   R6,R9                   ; DEDUCT FIELD
                   FB44  31  05AA          BRW     OVERFLOW                ; FIELD WIDTH OVERFLOW ERROR
```

```
                                 05AD          .SBTTL  HANDLER - Condition handler
                                 05AD  ;++
                                 05AD  ;
                                 05AD  ; FUNCTIONAL DESCRIPTION:
                                 05AD  ;
                                 05AD  ;       This condition handler is used to catch any errors which
                                 05AD  ;       ocurred while processing the arguments, such as access
                                 05AD  ;       violation.  This is because we don't want exceptions
                                 05AD  ;       occurring within the system service.
                                 05AD  ;       Care must be taken in this handler to deal with a second access
                                 05AD  ;       violation while storing the return value for $FAO.
                                 05AD  ;
                                 05AD  ; INPUTS:
                                 05AD  ;
                                 05AD  ;       CHF$L_SIGARGLST(AP) = Address of signal vector
                                 05AD  ;       CHF$L_MCHARGLST(AP) = Address of mechanism vector
                                 05AD  ;
                                 05AD  ; OUTPUTS:
                                 05AD  ;
                                 05AD  ;       The final R0 is set to the status code and the service
                                 05AD  ;       is exited via $UNWIND.
                                 05AD  ;--
                                 05AD
                                 05AD          .WEAK   EXE$SIGTORET
                                 05AD
                                 05AD  HANDLER:
                         0000    05AD          .WORD   0
                                 05AF
     6D    00000000'EF     9E    05AF          MOVAB   L^EXE$SIGTORET,(FP)     ;Simple handler for errors here
                   32      13    05B6          BEQL    90$                     ;*********
                                 05B8
                                 05B8          ASSUME  CHF$L_MCHARGLST,EQ,CHF$L_SIGARGLST+4
            50    04 AC    7D    05B8          MOVQ    CHF$L_SIGARGLST(AP),R0  ; Get address of signal argument list
  04 A0    00000920 8F     D1    05BC          CMPL    #SS$_UNWIND,CHF$L_SIG_NAME(R0) ;Unwinding?
                   24      13    05C4          BEQL    90$                     ;Exit if yes
                08 A1      D5    05C6          TSTL    CHF$L_MCH_DEPTH(R1)     ;Exception within FAO?
                   1A      12    05C9          BNEQ    80$                     ;Resignal if no
       0C A1    04 A0      D0    05CB          MOVL    CHF$L_SIG_NAME(R0),CHF$L_MCH_SAVR0(R1) ;Set final return status
                   7E      7C    05D0          CLRQ    -(SP)                   ;Clear depth and new PC arguments
      00000000'GF  02      FB    05D2          CALLS   #2,G^SYS$UNWIND         ;Unwind to establisher's caller
                                 05D9                                          ;***** The next instruction my ACCVIO
            50    08 AD    D0    05D9          MOVL    SF$L_SAVE_AP(FP),R0     ;Get address of FAO's argument list
            50    08 A0    D0    05DD          MOVL    OUTLEN(R0),R0           ;Output length requested?
                   02      13    05E1          BEQL    10$                     ;Branch if not
                   60      B4    05E3          CLRW    (R0)                    ;Indicate nothing returned in buffer
                                 05E5  10$:                                    ;***** End of potential ACCVIO
         50    0918 8F     3C    05E5  80$:    MOVZWL  #SS$_RESIGNAL,R0        ;Resignal (ignore after UNWIND)
                           04    05EA  90$:    RET                             ;
                                 05EB
                                 05EB
                                 05EB          .END
```

```
ARGCOUNT                              = 00000000              OP$_ADDF3             = 00000041
ASC_NAMES                               00000000 R      02    OP$_ADDG2             = 000040FD
CASE_BSB                                000000C8 R      02    OP$_ADDG3             = 000041FD
CASE_LOOP                               000000CA R      02    OP$_ADDH2             = 000060FD
CHF$C_MCHARGLST                       = 00000008              OP$_ADDH3             = 000061FD
CHF$L_MCH_DEPTH                       = 00000008              OP$_ADDP4             = 00000020
CHF$L_MCH_SAVR0                       = 0000000C              OP$_ADDP6             = 00000021
CHF$L_SIGARGLST                       = 00000004              OP$_ASHP              = 000000F8
CHF$L_SIG_NAME                        = 00000004              OP$_CLRD              = 0000007C
CNTRL_LENGTH                          = 00000010              OP$_CLRF              = 000000D4
CNTRL_TABLE                             00000010 R      02    OP$_CLRG              = 0000007C
CR                                    = 0000000D              OP$_CLRH              = 00007CFD
CVTASC                                  0000014B R      02    OP$_CMPD              = 00000071
CVTNUM                                  000001CC R      02    OP$_CMPF              = 00000051
CVT_BIN_TO_ASC                          0000025D R      02    OP$_CMPG              = 000051FD
DATA_TYPES                              00000024 R      02    OP$_CMPH              = 000071FD
DECR_ARGPTR                             000002AE R      02    OP$_CMPP3             = 00000035
DONE                                    000000FA R      02    OP$_CMPP4             = 00000037
ENDFIELD                                000002EF R      02    OP$_CRC               = 0000000B
EXCL                                  = 00000021              OP$_CVTBD             = 0000006C
EXE$FAO                                 0000003F RG     02    OP$_CVTBF             = 0000004C
EXE$FAOL                                0000004C RG     02    OP$_CVTBG             = 00004CFD
EXE$SIGTORET                            ********W GX    02    OP$_CVTBH             = 00006CFD
FAO                                     00000057 R      02    OP$_CVTDB             = 00000068
FAO_CASE                                000000CF R      02    OP$_CVTDF             = 00000076
FAO_EXIT                                000000FD R      02    OP$_CVTDH             = 000032FD
FF                                    = 0000000C              OP$_CVTDL             = 0000006A
FIELDEND                              = FFFFFFFC              OP$_CVTDW             = 00000069
FIELDS                                  0000002C R      02    OP$_CVTFB             = 00000048
FIRSTARG                              = 00000010              OP$_CVTFD             = 00000056
GETCHAR                                 00000109 R      02    OP$_CVTFG             = 000099FD
GETCOUNT                                00000116 R      02    OP$_CVTFH             = 000098FD
HANDLER                                 000005AD R      02    OP$_CVTFL             = 0000004A
ID_REG_MASK                           = 000001FC              OP$_CVTFW             = 00000049
ILLEGAL                                 000000EC R      02    OP$_CVTGB             = 000048FD
ILLFIELD                                000002E1 R      02    OP$_CVTGF             = 000033FD
INCR_ARGPTR                             000002AB R      02    OP$_CVTGH             = 000056FD
INDSC                                 = 00000004              OP$_CVTGL             = 00004AFD
INLEN                                 = FFFFFFF0              OP$_CVTGW             = 000049FD
INPTR                                 = FFFFFFF4              OP$_CVTHB             = 000068FD
INSERT_CHAR                             000002B9 R      02    OP$_CVTHD             = 0000F7FD
INSERT_IT                               000002BF R      02    OP$_CVTHF             = 0000F6FD
INSERT_OVF                              000002BC R      02    OP$_CVTHG             = 000076FD
LASTVAL                               = FFFFFFF8              OP$_CVTHL             = 00006AFD
LF                                    = 0000000A              OP$_CVTHW             = 000069FD
MAIN_SCAN                               00000064 R      02    OP$_CVTLD             = 0000006E
NEWLINE                                 000002B1 R      02    OP$_CVTLF             = 0000004E
OCT_HEX_DIGITS                          00000033 R      02    OP$_CVTLG             = 00004EFD
ONECHAR_INDEX                         = 00000008              OP$_CVTLH             = 00006EFD
ONE_CHAR_CNTRLS                         00000018 R      02    OP$_CVTLP             = 000000F9
OP$_ACBD                              = 0000006F              OP$_CVTPL             = 00000036
OP$_ACBF                              = 0000004F              OP$_CVTPS             = 00000008
OP$_ACBG                              = 00004FFD              OP$_CVTPT             = 00000024
OP$_ACBH                              = 00006FFD              OP$_CVTRDL            = 0000006B
OP$_ADDD2                             = 00000060              OP$_CVTRFL            = 0000004B
OP$_ADDD3                             = 00000061              OP$_CVTRGL            = 00004BFD
OP$_ADDF2                             = 00000040              OP$_CVTRHL            = 00006BFD
```

```
                                                 K 9
SYSFAO                          - FORMATTED ASCII OUTPUT SYSTEM SERVICE  16-SEP-1984 02:06:18  VAX/VMS Macro V04-00        Page  30
Symbol table                                                              5-SEP-1984 03:53:14  [SYS.SRC]SYSFAO.MAR;1             (10)

OP$_CVTSP                       = 00000009                   OP$_SUBP6                       = 00000023
OP$_CVTTP                       = 00000026                   OP$_TSTD                        = 00000073
OP$_CVTWD                       = 0000006D                   OP$_TSTF                        = 00000053
OP$_CVTWF                       = 0000004D                   OP$_TSTG                        = 000053FD
OP$_CVTWG                       = 00004DFD                   OP$_TSTH                        = 000073FD
OP$_CVTWH                       = 00006DFD                   OUTDSC                          = 0000000C
OP$_DIVD2                       = 00000066                   OUTLEN                          = 00000008
OP$_DIVD3                       = 00000067                   OVERFLOW                          000000F1 R       02
OP$_DIVF2                       = 00000046                   PARSE_DIRECTIVE                   0000008F R       02
OP$_DIVF3                       = 00000047                   PERCENT                           00000301 R       02
OP$_DIVG2                       = 000046FD                   PERCENT_STR                       00000027 R       02
OP$_DIVG3                       = 000047FD                   RADIX                             0000003A R       02
OP$_DIVH2                       = 000066FD                   REPEATIT                          000002C6 R       02
OP$_DIVH3                       = 000067FD                   REPLACEMENT                       0000002F R       02
OP$_DIVP                        = 00000027                   REPLACE_CHRS                      0000001C R       02
OP$_EDITPC                      = 00000038                   REPL_OFFSET                     = 0000000C
OP$_EMODD                       = 00000074                   SF$L_SAVE_AP                    = 00000008
OP$_EMODF                       = 00000054                   SS$_BADPARAM                    = 00000014
OP$_EMODG                       = 000054FD                   SS$_BUFFEROVF                   = 00000601
OP$_EMODH                       = 000074FD                   SS$_NORMAL                      = 00000001
OP$_MATCHC                      = 00000039                   SS$_RESIGNAL                    = 00000918
OP$_MNEGD                       = 00000072                   SS$_UNWIND                      = 00000920
OP$_MNEGF                       = 00000052                   STARTFIELD                        000002DD R       02
OP$_MNEGG                       = 000052FD                   STARTOK                           000002E4 R       02
OP$_MNEGH                       = 000072FD                   STRING_TYPES                      00000020 R       02
OP$_MOVD                        = 00000070                   SYS$ASCTIM                        ********    GX   02
OP$_MOVF                        = 00000050                   SYS$IDTOASC                       ********    GX   02
OP$_MOVG                        = 000050FD                   SYS$UNWIND                        ********     X   02
OP$_MOVH                        = 000070FD                   TAB                             = 00000009
OP$_MOVP                        = 00000034                   TWO_CHAR_CNTRLS                   00000010 R       02
OP$_MOVTC                       = 0000002E                   UIC$K_ID_FORMAT                 = 00000002
OP$_MOVTUC                      = 0000002F                   UIC$K_UIC_FORMAT                = 00000000
OP$_MULD2                       = 00000064                   UIC$K_WILD_GROUP                = 00003FFF
OP$_MULD3                       = 00000065                   UIC$K_WILD_MEMBER               = 0000FFFF
OP$_MULF2                       = 00000044                   UIC$S_FORMAT                    = 00000002
OP$_MULF3                       = 00000045                   UIC$S_GROUP                     = 0000000E
OP$_MULG2                       = 000044FD                   UIC$S_MEMBER                    = 00000010
OP$_MULG3                       = 000045FD                   UIC$V_FORMAT                    = 0000001E
OP$_MULH2                       = 000064FD                   UIC$V_GROUP                     = 00000010
OP$_MULH3                       = 000065FD                   UIC$V_MEMBER                    = 00000000
OP$_MULP                        = 00000025
OP$_POLYD                       = 00000075
OP$_POLYF                       = 00000055
OP$_POLYG                       = 000055FD
OP$_POLYH                       = 000075FD
OP$_SCANC                       = 0000002A
OP$_SKPC                        = 0000003B
OP$_SPANC                       = 0000002B
OP$_SUBD2                       = 00000062
OP$_SUBD3                       = 00000063
OP$_SUBF2                       = 00000042
OP$_SUBF3                       = 00000043
OP$_SUBG2                       = 000042FD
OP$_SUBG3                       = 000043FD
OP$_SUBH2                       = 000062FD
OP$_SUBH3                       = 000063FD
OP$_SUBP4                       = 00000022
```

```
                                              +----------------+
                                              ! Psect synopsis !
                                              +----------------+

PSECT name                      Allocation           PSECT No.  Attributes
----------                      ----------           ---------  ----------
.  ABS  .                       00000000  (     0.)  00 (  0.)  NOPIC    USR    CON    ABS    LCL NOSHR NOEXE NORD   NOWRT NOVEC BYTE
$ABS$                           00000000  (     0.)  01 (  1.)  NOPIC    USR    CON    ABS    LCL NOSHR   EXE   RD     WRT NOVEC BYTE
YF$SYSFAO                       000005EB  (  1515.)  02 (  2.)  NOPIC    USR    CON    REL    LCL NOSHR   EXE   RD     WRT NOVEC BYTE

                                           +------------------------+
                                           ! Performance indicators !
                                           +------------------------+

Phase                      Page faults    CPU Time       Elapsed Time
-----                      -----------    --------       ------------
Initialization                      35    00:00:00.09    00:00:00.95
Command processing                 140    00:00:00.78    00:00:08.10
Pass 1                             497    00:00:17.29    00:00:45.32
Symbol table sort                    0    00:00:01.48    00:00:06.27
Pass 2                             234    00:00:05.88    00:00:16.94
Symbol table output                 25    00:00:00.19    00:00:01.16
Psect synopsis output                2    00:00:00.02    00:00:00.02
Cross-reference output               0    00:00:00.00    00:00:00.00
Assembler run totals               935    00:00:25.74    00:01:18.76

The working set limit was 2100 pages.
81515 bytes (160 pages) of virtual memory were used to buffer the intermediate code.
There were 60 pages of symbol table space allocated to hold 892 non-local and 90 local symbols.
4147 source lines were read in Pass 1, producing 16 object records in Pass 2.
143 pages of virtual memory were used to define 142 macros.

                                          +----------------------------+
                                          ! Macro library statistics !
                                          +----------------------------+

Macro library name                         Macros defined
------------------                         --------------
_$255$DUA28:[SYS.OBJ]LIB.MLB;1                         1
_$255$DUA28:[SYSLIB]STARLET.MLB;2                     12
TOTALS (all libraries)                                13

946 GETS were required to define 13 macros.

There were no errors, warnings or information messages.

MACRO/LIS=LIS$:SYSFAO/OBJ=OBJ$:SYSFAO MASD$:[EMULAT.SRC]MISSING/UPDATE=(MASD$:[EMULAT.ENH]MISSING)+MASD$:[SYS.SRC]SYSFAO/UPDATE=(MAS
```

0384 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY

SYSGETJPI
LIS

SYSERAPAT
LIS

SYSFAO
LIS

SYSGETDVI
LIS

SYSEXIT
LIS

SYSEVTSRV
LIS

SYSFORCEX
LIS